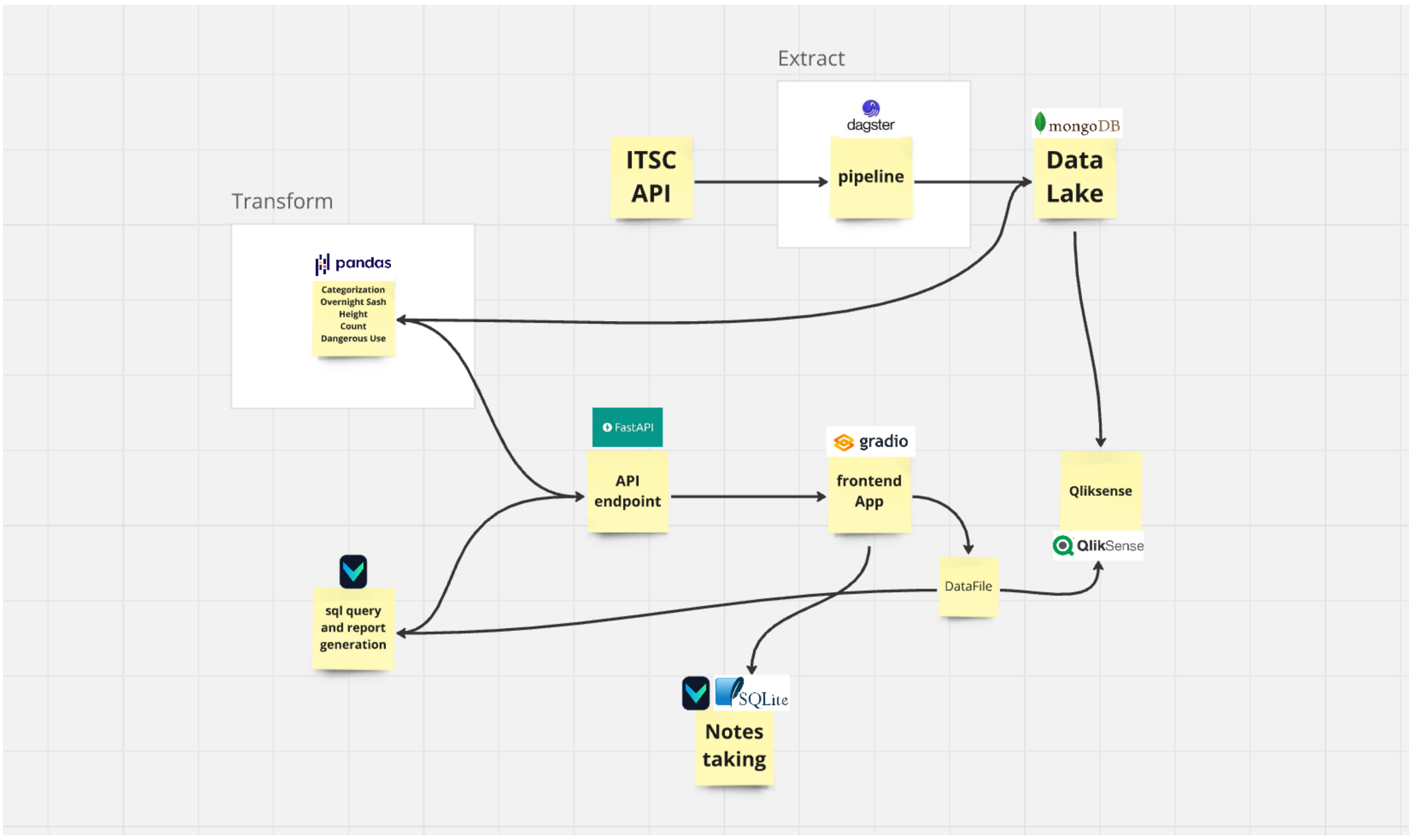
SUST PROJECT fume hood full version

workflow:



```
Dagster(completed)
Base on the Api.py, Cleaner.py and DataCaller.py to run up a pipline orchestrator
fetch data regularly per hour serve as a copy from the ITSC Elastic Search database
The data would store to MongoDB
access GUI through localhost:3000
**Documentation for `__init__.py`**
=====================================

**Overview**
------------

This file defines a set of assets and resources for a Dagster pipeline. It imports necessary
libraries, defines functions for data processing and database interaction, and creates assets
for data ingestion and processing.

**Imported Libraries**
----------------------

* `dagster`: for defining assets and resources
* `pandas`: for data manipulation and analysis
* `fumehood_project.Cleaner`: for data cleaning functions
* `fumehood_project.DataCaller`: for data retrieval functions
* `pymongo`: for interacting with MongoDB

**Functions**
-------------
```

```
### `ust_api_raw_by_instance(instance)`

* Returns the latest data for a given instance from the `latest` function in `fumehood_projec
t.DataCaller`.
* If the data is empty, returns an empty DataFrame.

### `ust_api_raw()`

* Creates a list of data for each instance in the `fumehood_list.csv` file.
* Concatenates the data for each instance using `pd.concat`.
* Returns the concatenated data.

### `ust_api_clean(ust_api_raw)`

* Applies a series of data cleaning functions to the `ust_api_raw` data:
    + `transform`: applies a transformation function from `fumehood_project.Cleaner`.
    + `date_transform`: applies a date transformation function from `fumehood_project.Cleaner
`.
    + `period`: applies a period function from `fumehood_project.Cleaner`.
    + `merge`: merges the data using `pd.merge`.
    + `sort_to_time`: sorts the data by time using `pd.sort_values`.
    + `format`: formats the data using `fumehood_project.Cleaner`.
* Returns the cleaned data.

### `type_control(df)`

* Converts the columns of a DataFrame to specified data types using `pd.astype`.
* Returns the DataFrame with the converted columns.

### `get_database(df)`

* Connects to a MongoDB database using `pymongo`.
* Retrieves data from the database for each instance in the `df` DataFrame.
* Returns the retrieved data.

### `filter_exist(df_latest, df_exist)`

* Concatenates the `df_latest` and `df_exist` DataFrames.
* Drops duplicates using `pd.drop_duplicates`.
* Returns the concatenated DataFrame with duplicates removed.

### `insert_to_database(df)`

* Connects to a MongoDB database using `pymongo`.
* Inserts the data from the `df` DataFrame into the database.
* Returns a success message.

### `api_to_database(ust_api_clean)`

* Calls the `type_control` function to convert the columns of the `ust_api_clean` data.
* Calls the `get_database` function to retrieve data from the database.
* Calls the `filter_exist` function to remove duplicates from the data.
* Calls the `insert_to_database` function to insert the data into the database.
* Returns a success message if the data is inserted successfully.

**Assets**
----------
```

```
* `ust_api_raw`: an asset that creates a list of data for each instance in the `fumehood_lis
t.csv` file.
* `ust_api_clean`: an asset that applies a series of data cleaning functions to the `ust_api_
raw` data.
* `api_to_database`: an asset that inserts the cleaned data into the database.

**Resources**
-------------

* `partitions_def`: a resource that defines the partitions for the `ust_api_raw` asset.
* `retry_policy`: a resource that defines the retry policy for the `ust_api_raw` and `ust_api
_clean` assets.
```

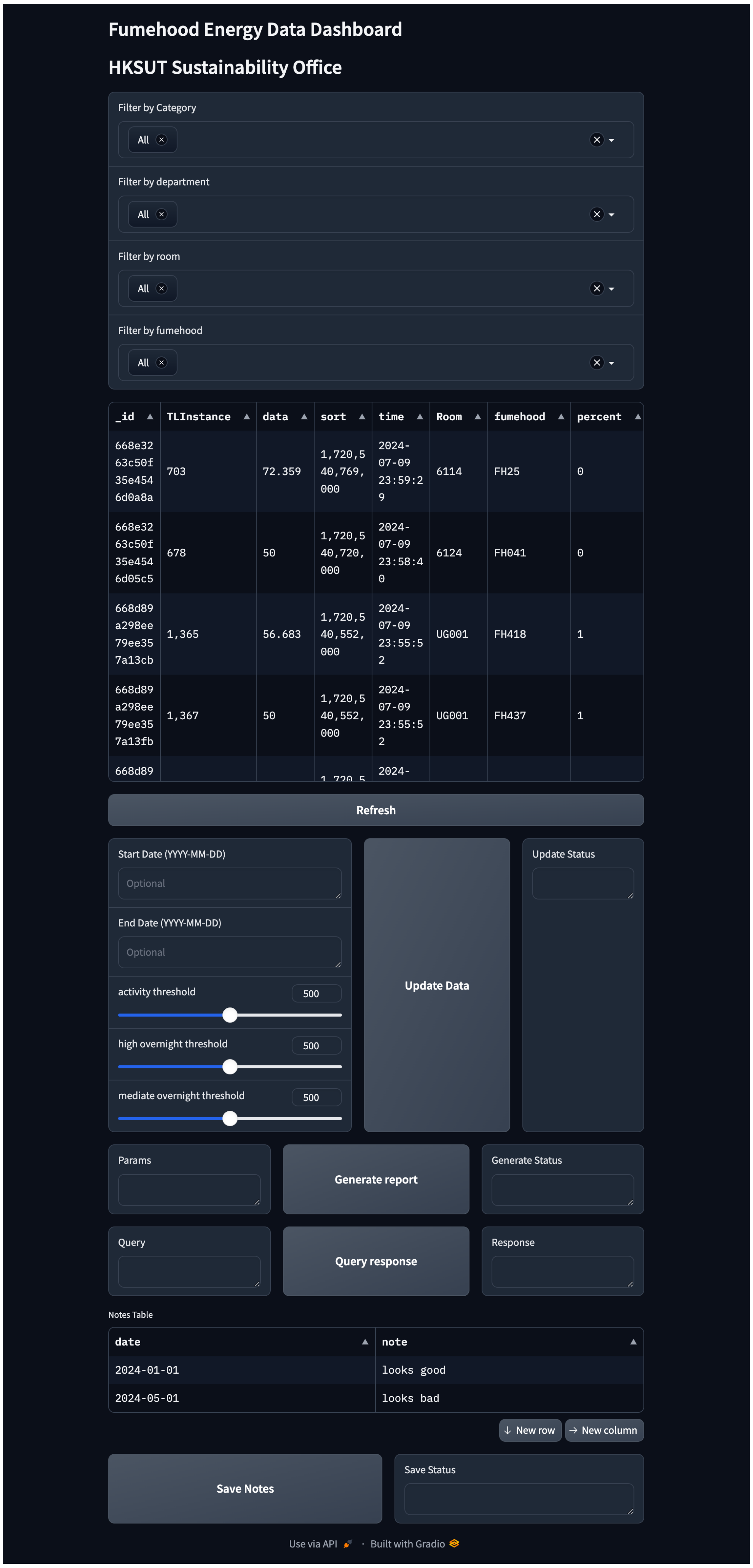
```
MongoDB(completed)
Save all the data from Dagster
Deployed with Docker
Mongodb (Database) access through localhost:80 also expose to public with 143.89.6.90:80
For stability, managed by docker
Credentials:
account admin, password ilovenetzero
```

```
Pandas Backend Calculation(completed)
Same as Model.py, VirtualPlot.py in fumehoodv2
serve the calculate through FastAPI to frontend gradio
```

```
Gradio(finished draft)
**Gradio App Documentation**
==========================

**Overview**
------------

This Gradio app is a data dashboard for Fumehood Energy Data. It allows users to filter and v
isualize data based on various parameters.

**Components**
--------------

### 1. Leaderboard

*   **Render Leaderboard**: Displays the leaderboard markdown.
*   **Create Leaderboard**: Filters the DataFrame based on selected filters and returns the s
orted DataFrame.
*   **Leaderboard Table**: Displays the leaderboard table.

### 2. Data Update

*   **Update Data**: Fetches data from the database based on input parameters and updates the
DataFrame.
*   **Start Date Input**: Input field for start date (YYYY-MM-DD).
*   **End Date Input**: Input field for end date (YYYY-MM-DD).
*   **Activity Threshold**: Slider for activity threshold (0-1000).
*   **High Overnight Threshold**: Slider for high overnight threshold (0-1000).
*   **Mediate Overnight Threshold**: Slider for mediate overnight threshold (0-1000).
*   **Update Button**: Button to update the data.
```

```
### 3. Report Generation

*   **Generate Report**: Button to generate a report based on input parameters.
*   **Report Input**: Input field for report parameters.
*   **Report Message**: Output field for report status.

### 4. Query Response

*   **Query Response**: Button to query the response based on input parameters.
*   **Query Input**: Input field for query parameters.
*   **Query Message**: Output field for query response.

### 5. Notes Table

*   **Notes Table**: Displays the notes table.
*   **Save Notes**: Button to save the notes table.
*   **Output Message**: Output field for save status.

**Functions**
-------------

### 1. `fetch_data`

*   Fetches data from the database based on input parameters.
*   Returns the cleaned DataFrame.

### 2. `create_leaderboard`

*   Filters the DataFrame based on selected filters and returns the sorted DataFrame.

### 3. `update_data`

*   Updates the DataFrame based on input parameters.

### 4. `report_generate`

*   Generates a report based on input parameters.

### 5. `query_generate`

*   Queries the response based on input parameters.

### 6. `read_notes_table`

*   Reads the notes table from the CSV file.

### 7. `save_notes_table`

*   Saves the notes table to the CSV file.

### 8. `update_and_save_notes`

*   Updates and saves the notes table.

**Notes**
--------
```

```
*   The app uses Gradio for building the interface.
*   The data is stored in a MongoDB database.
*   The app uses pandas for data manipulation and analysis.
*   The app uses requests for API calls.
*   The app uses a custom `Model` class for data cleaning and processing.
*   The app uses a custom `Cleaner` class for data validation.
*   The app uses a custom `VirtualPlot` class for data visualization.
```



Fumehood Energy Data Dashboard

HKSUT Sustainability Office

Filter by Category: All

Filter by department: All

Filter by room: All

Filter by fumehood: All

_id	TLInstance	data	sort	time	Room	fumehood	percent
668e3263c50f35e4546d0a8a	703	72.359	1,720,540,769,000	2024-07-09 23:59:29	6114	FH25	0
668e3263c50f35e4546d05c5	678	50	1,720,540,720,000	2024-07-09 23:58:40	6124	FH041	0
668d89a298ee79ee357a13cb	1,365	56.683	1,720,540,552,000	2024-07-09 23:55:52	UG001	FH418	1
668d89a298ee79ee357a13fb	1,367	50	1,720,540,552,000	2024-07-09 23:55:52	UG001	FH437	1
668d89			1,720,5	2024-			

Refresh

Start Date (YYYY-MM-DD): Optional

End Date (YYYY-MM-DD): Optional

activity threshold: 500

high overnight threshold: 500

mediate overnight threshold: 500

Update Data

Update Status

Params

Generate report

Generate Status

Query

Query response

Response

Notes Table

date	note
2024-01-01	looks good
2024-05-01	looks bad

New row · New column

Save Notes

Save Status

Use via API · Built with Gradio

```
Vanna SQL query(pending to implement)
Allow user to get relevant data and plot with natural english language.
Finished testing functionalities, result is in vanna_testing.ipynb.
```

```
Note taking (SQLite, Vanna)(pending to implement)
storing different fumehoods notes support CRUD and hope to have retreive report ability.
```

```
FastAPI(pending to migrate from flask)
migrate functions from flask except the frontend components.
serve backend endpoints for pandas backend and vanna.
```